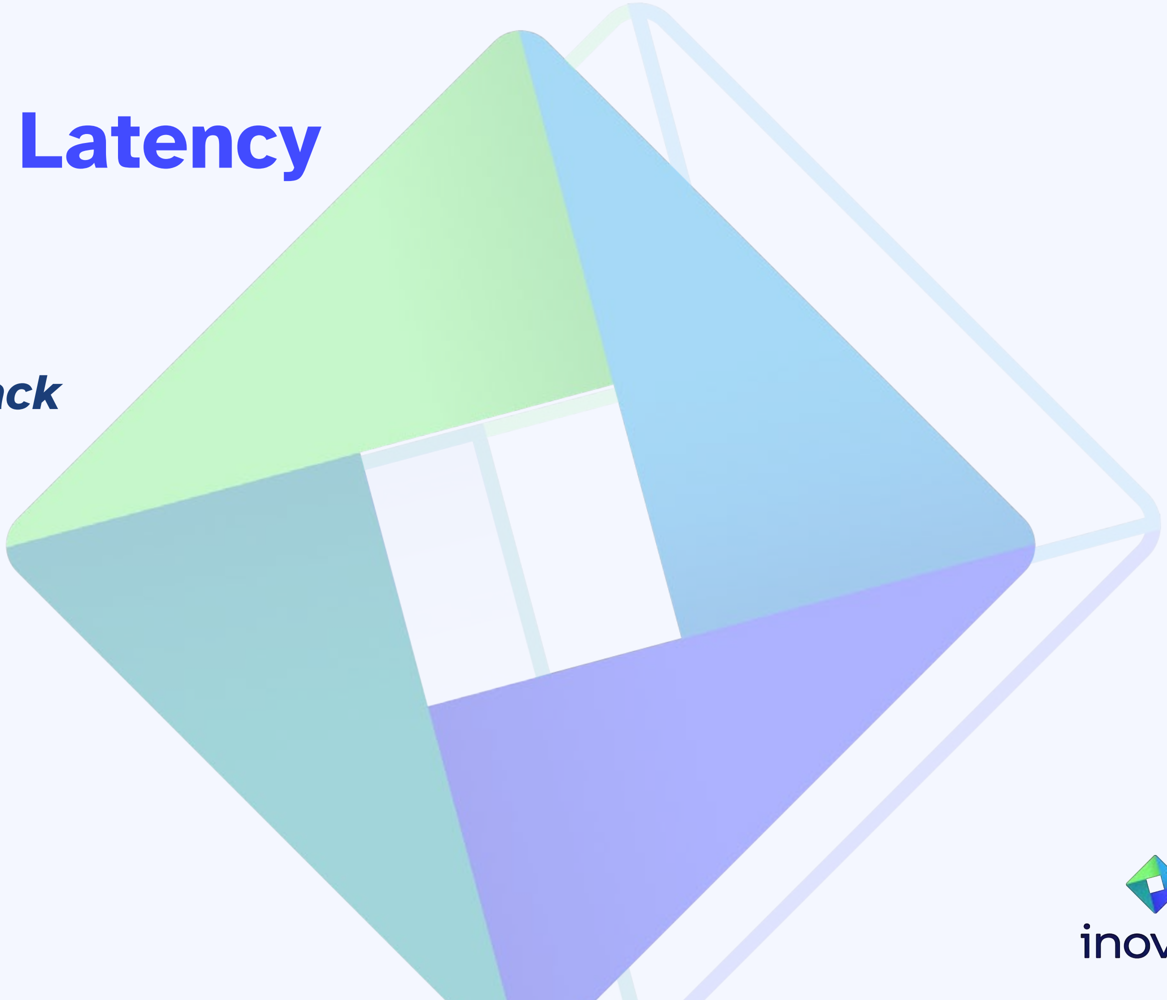
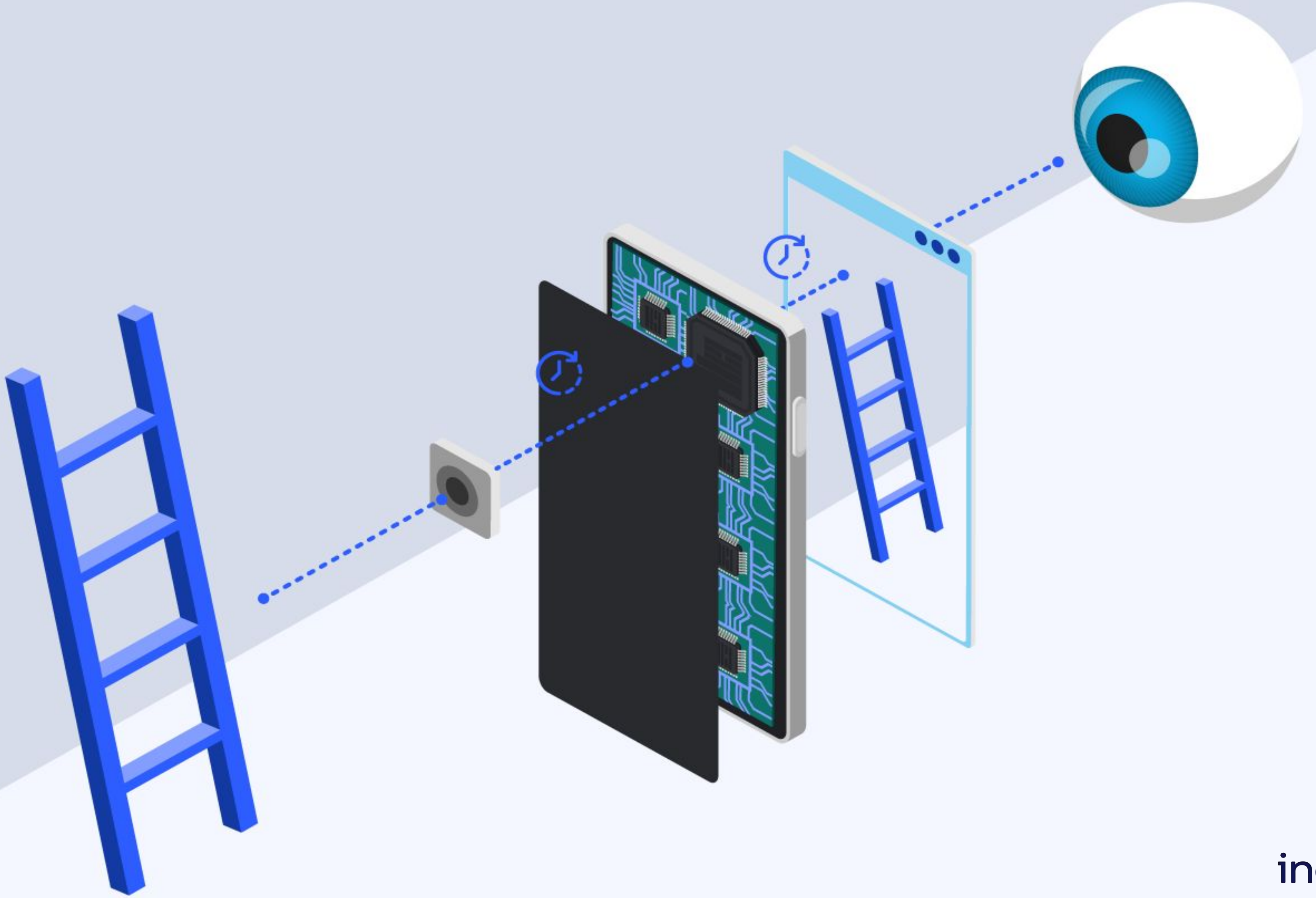
Glass-To-Glass Latency in Android

How to debug the Android Graphics Stack

Stefan Lengfeld
March 12th, 2024 · Munich



inovex



inovex

Stefan Lengfeld

stefan.lengfeld
@inovex.de

lengfeld

stefan.lengfeld.xyz

Android and Linux Embedded Developer

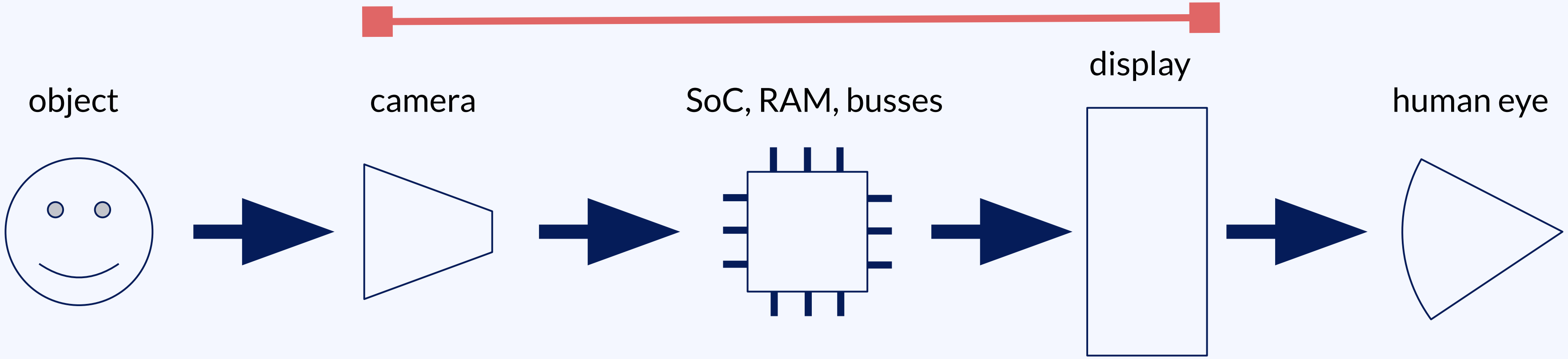
- 7+ years at inovex
- 9+ years professional embedded software development
- many more years a linux enthusiast

Main Topics

- Embedded Systems (Linux and Android)
- Linux Kernel
- Build systems
- Linux Graphics Stack

inovex

What is the glass-to-glass latency?



inovex

Why is the glass to glass latency important?

Augmented reality (AR)

Virtual reality (VR)

Mixed reality (MR)

Configurations

- semi transparent displays
- non transparent displays with real world camera blending

inovex

Test device

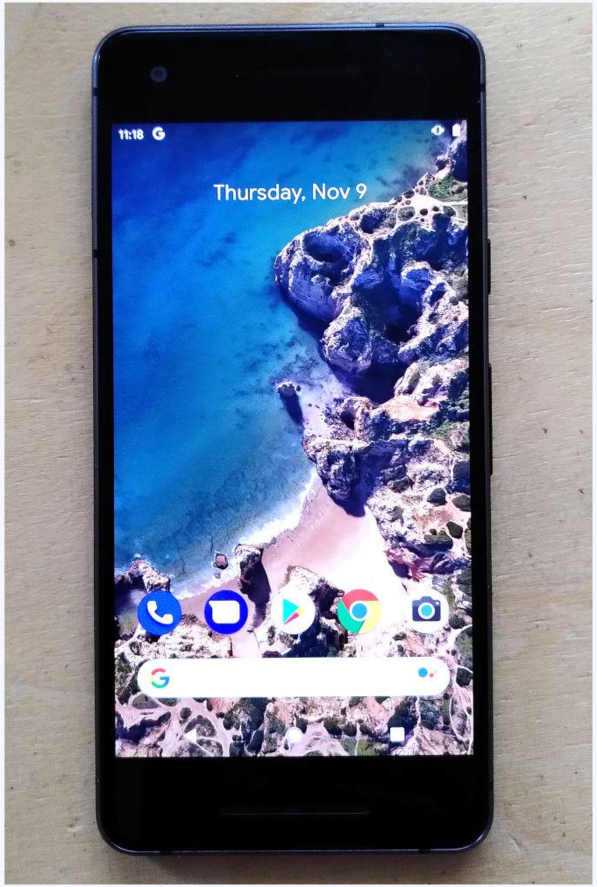
Pixel 2 from Google

- first release 2017
- discontinued 2019

But seems outdated,
but the result are still the same
with current hardware.

For my tests:

- Display 60 Hz => 16 ms
- Camera 30 Hz => 33 ms



inovex

How to measure the glass-to-glass latency? Not!

real stopwatch:

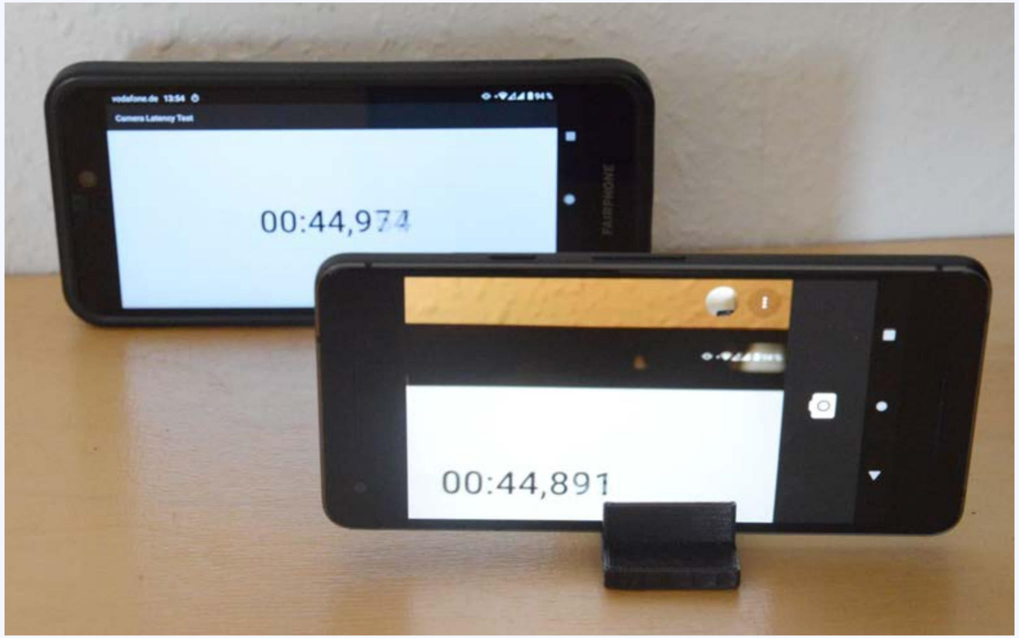
44,974 ms

image of stopwatch:

44. 891 ms

difference: **83 ms**

But: Is this correct?



Yes, it’s “correct”!

The more precise answer is 57 - 106 ms.

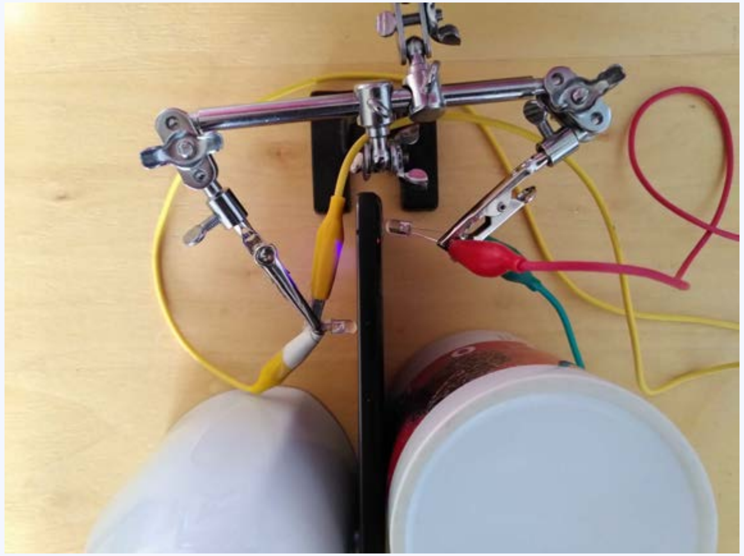
But why?

Test equipment

LEDs, photo-diodes, cables, resistors, coffee mugs, power supply

Oscilloscope, arduino

Android's **systrace**



inovex

What is systrace? The Android system tracer!

It’s build upon the Linux kernels *ftrace* tracing framework!

It includes a bunch of kernel events:

processes, scheduler events, irqs, driver subsystem events

And a lot of events from the Android userspace!

And you can use it in our own application:

- Java/Kotlin:
  `Trace.beginSection(...)`   `Trace.endSection()`
- C/C++:
  `ATrace_beginSection(...)`   `ATrace_endSection()`

How to use systrace - Part 1

On the commandline:

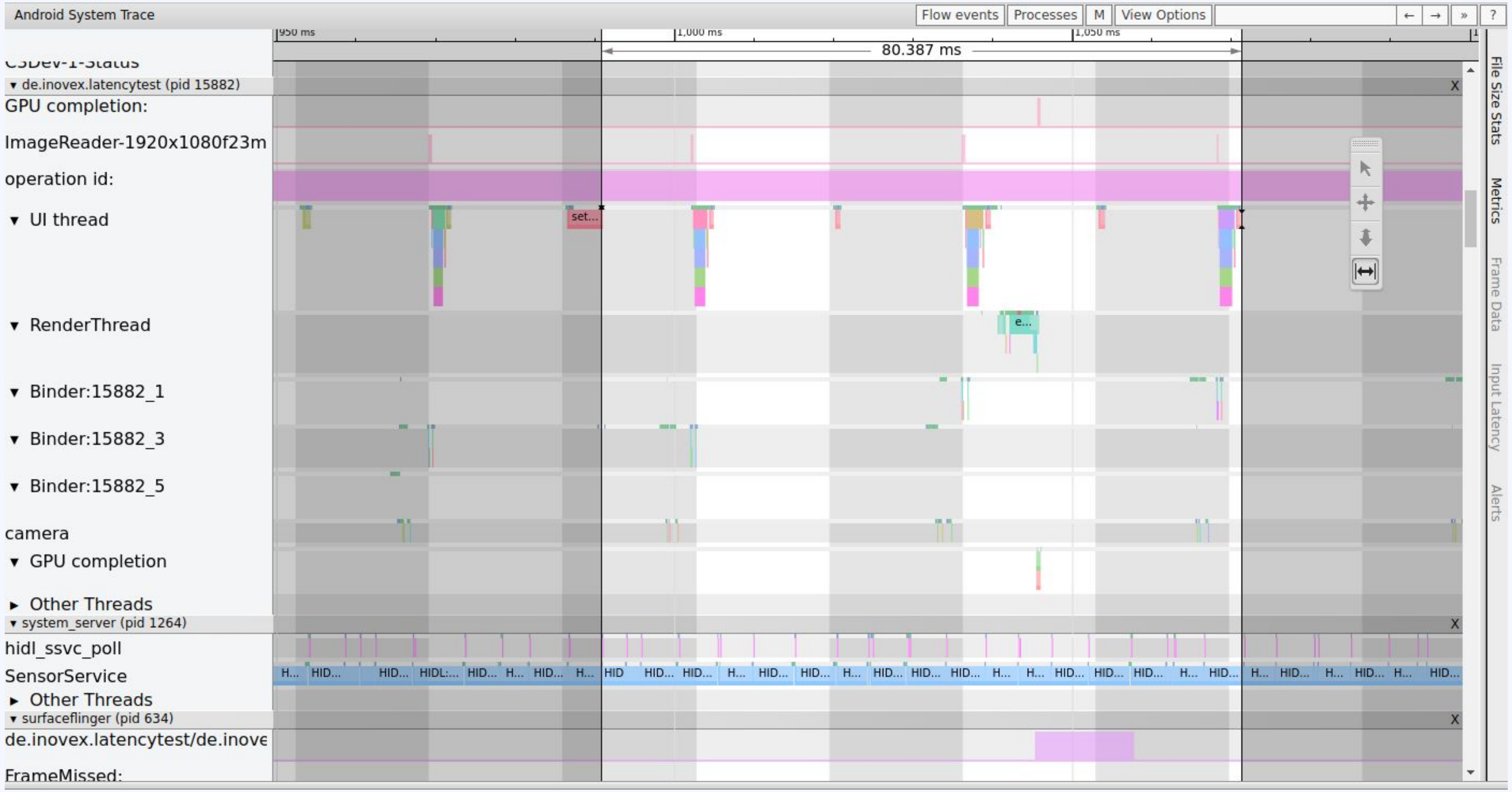
```
$ $HOME/Sdk/platform-tools/systrace/systrace.py \
    --atrace-categories=sched,gfx,hal,irq,ion,camera,sm \
    --time=2 \
    -o systrace.html \
    -a de.inovex.latencytest

$ $BROWSER systrace.html
```

https://perfetto.dev/ (Next gen version of systrace)
https://developer.android.com/topic/performance/tracing/
https://source.android.com/docs/core/tests/debug/systrace

inovex

How to use systrace - Part 2



After

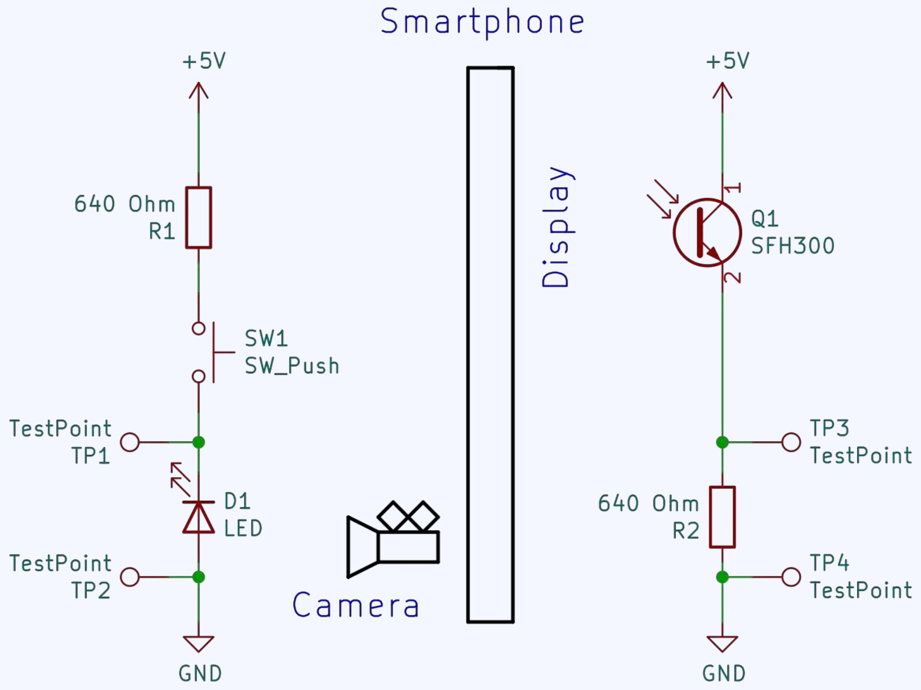
four weeks

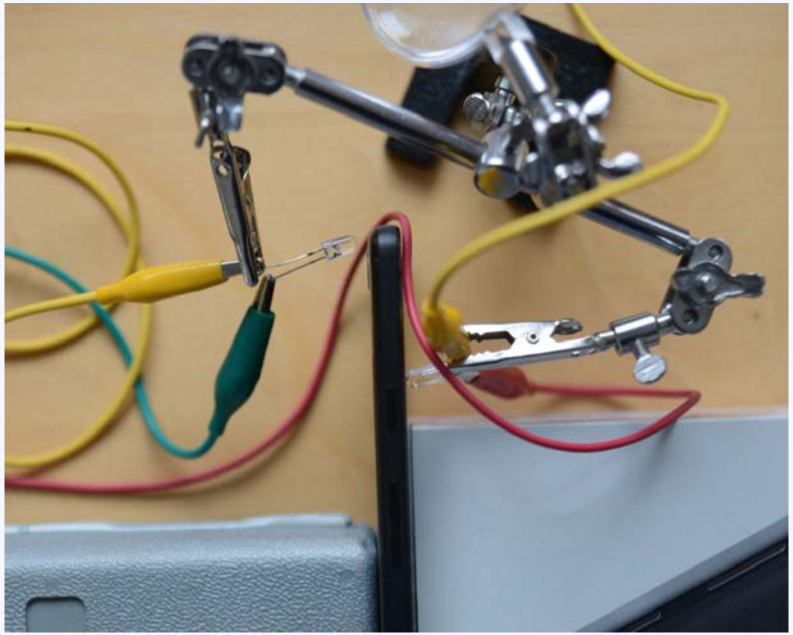
~15 tests

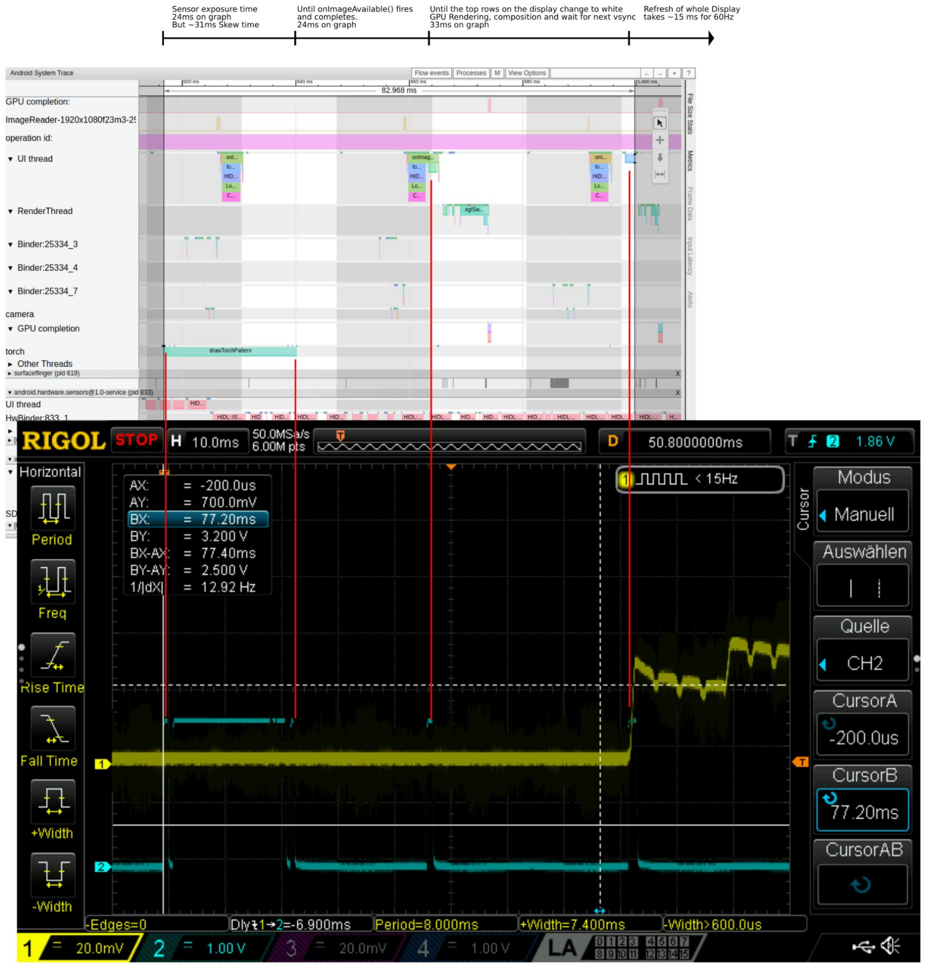
a bit of cabling

a lot of system traces

I got the following setup:


Smartphone
+5V
640 Ohm
R1
SW1
SW_Push
TestPoint
TP1
D1
LED
TestPoint
TP2
GND
Camera
Display
+5V
Q1
SFH300
TP3
TestPoint
640 Ohm
R2
TP4
TestPoint
GND





```
AX:    =  -200.0us
AY:    =  700.0mV
BX:    =  77.20ms
BY:    =  3.200 V
BX-AX: =  77.40ms
BY-AY: =  2.500 V
1/|dX| =  12.92 Hz
```

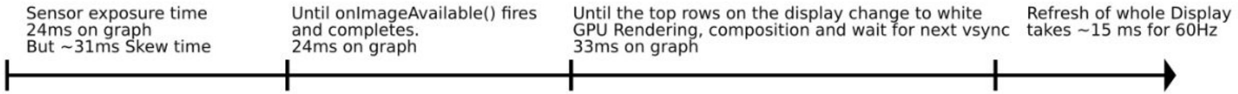
Graphic shows

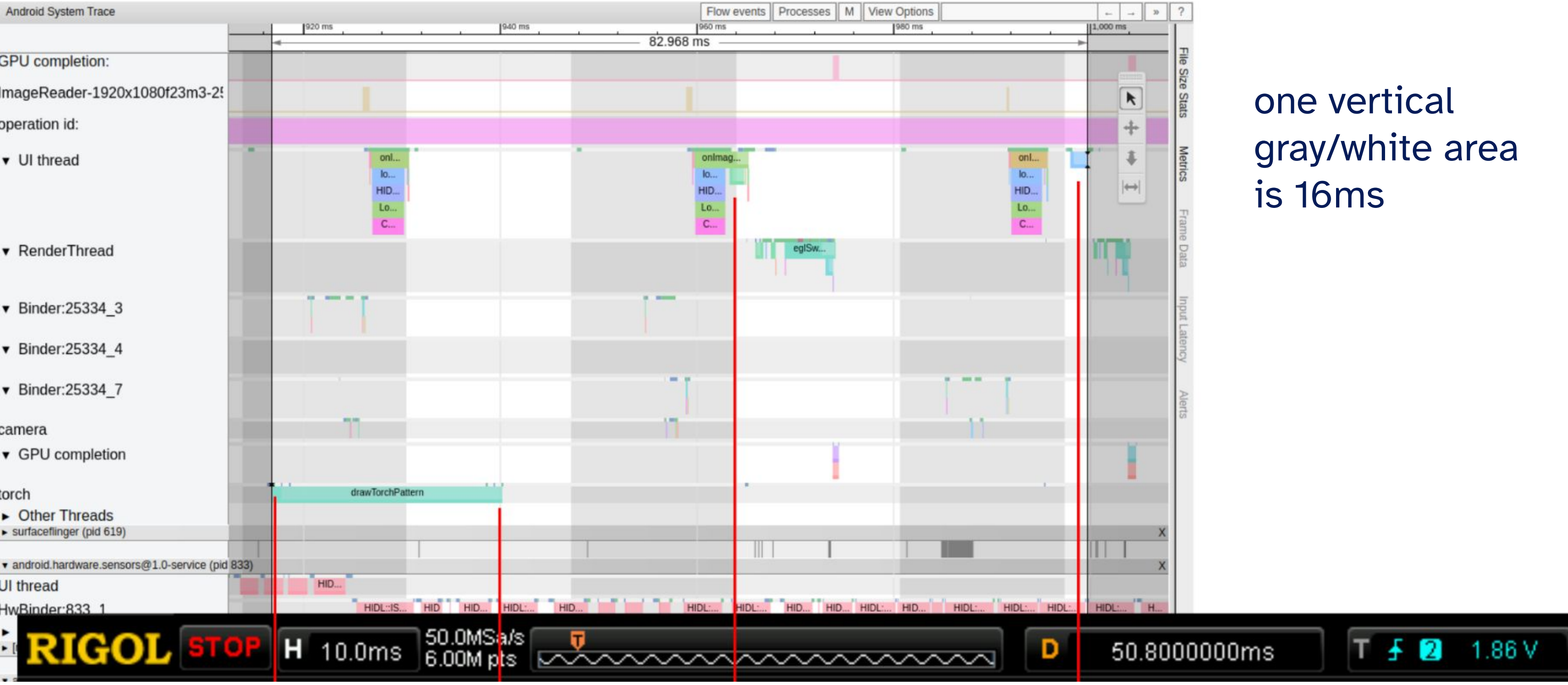
- a systrace graph and
- a oscilloscope capture

which are lined up.

Blue line/voltage:
LED and markes

Yellow line/voltage:
photodiode





one vertical gray/white area is 16ms

Components of the glass-to-glass latency

Timings from the systrace graph and oscilloscope capture:

0-33 ms	camera sensor exposure
24 ms	sensor to app (onImageAvailable Callback)
~33 ms	app to display (GPU, surfaceflinger, vsync)
0-16 ms	display scanout

→ The latency is **57 to 106** ms (min, max).

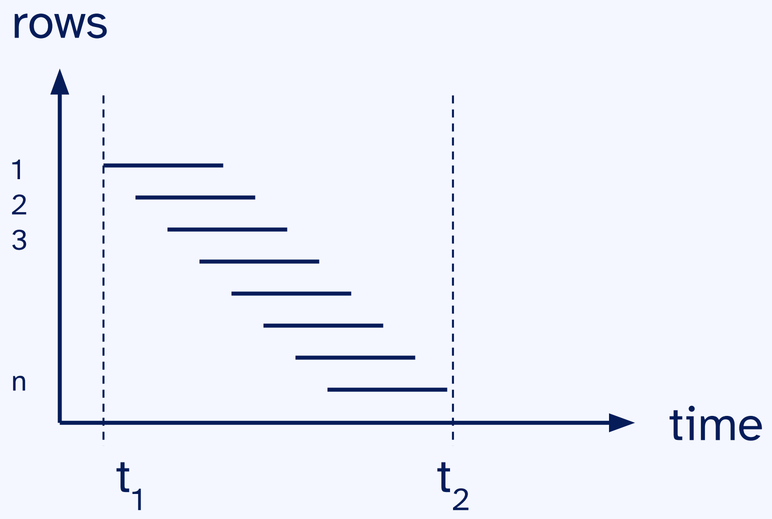
The details:

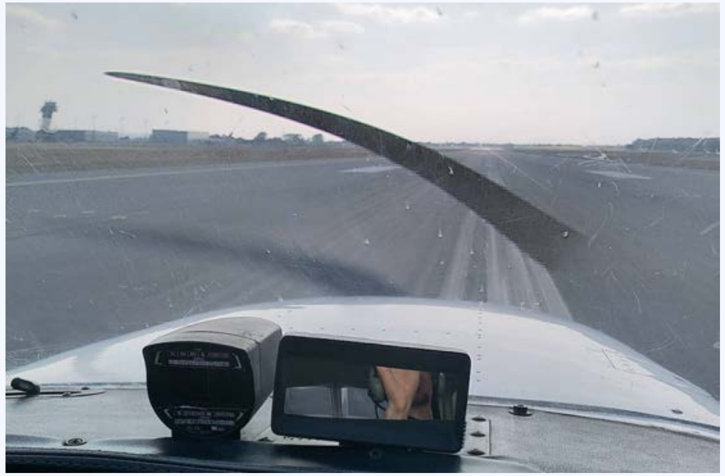
- Camera sensor
  - Rolling shutter
  - Skew time
- Display scanout
- Rendering and vsyncs

inovex

Sensor - Rolling shutter effect

The sensor does not exposure all pixels at the same time.
The exposure starts row after row.





Results in *kind of* motion blur like from digital or analog with a mechanical shutter camera but different.

$$t_2 - t_1 := \text{skew time}$$

inovex

Sensor - Skew time

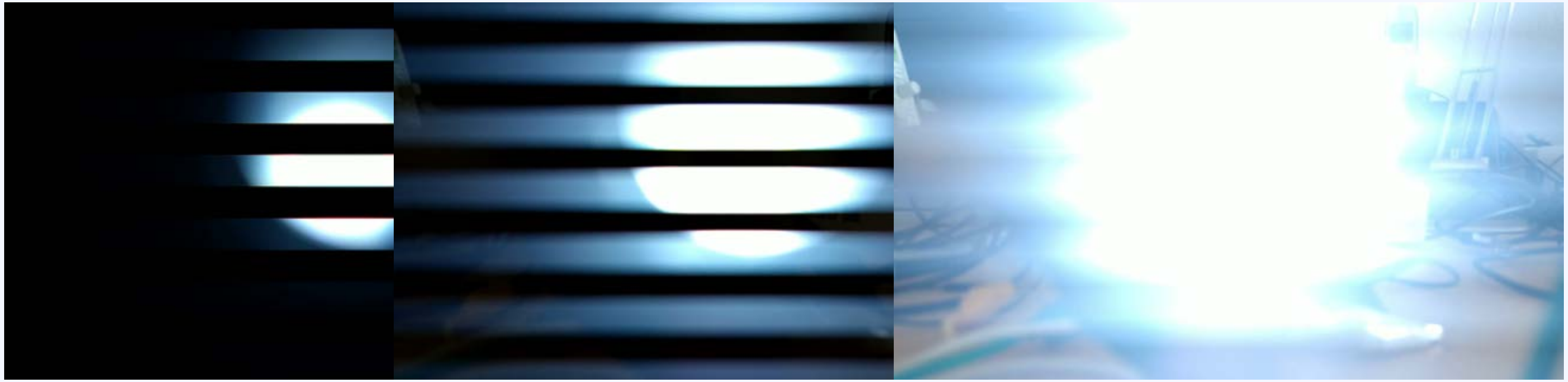
The time between the exposure
of the first and the last row
does **not** depend on the exposure time.

The skew time is always 32 ms

```
#define LED 4   // pin of the LED

void setup() {
  pinMode(LED, OUTPUT); // Declare an output
}

void loop() {
  digitalWrite(LED, HIGH); // Turn the LED on
  delayMicroseconds(2000); // wait 2 ms
  digitalWrite(LED, LOW);  // Turn the LED off
  delayMicroseconds(2000); // wait 2 ms
}
```

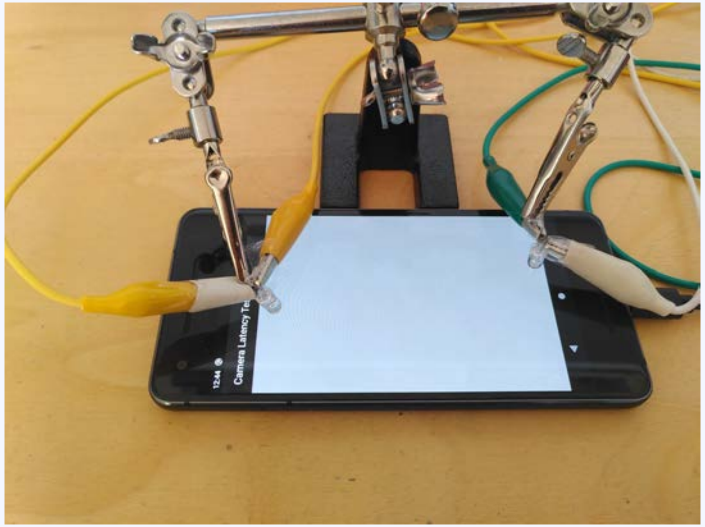


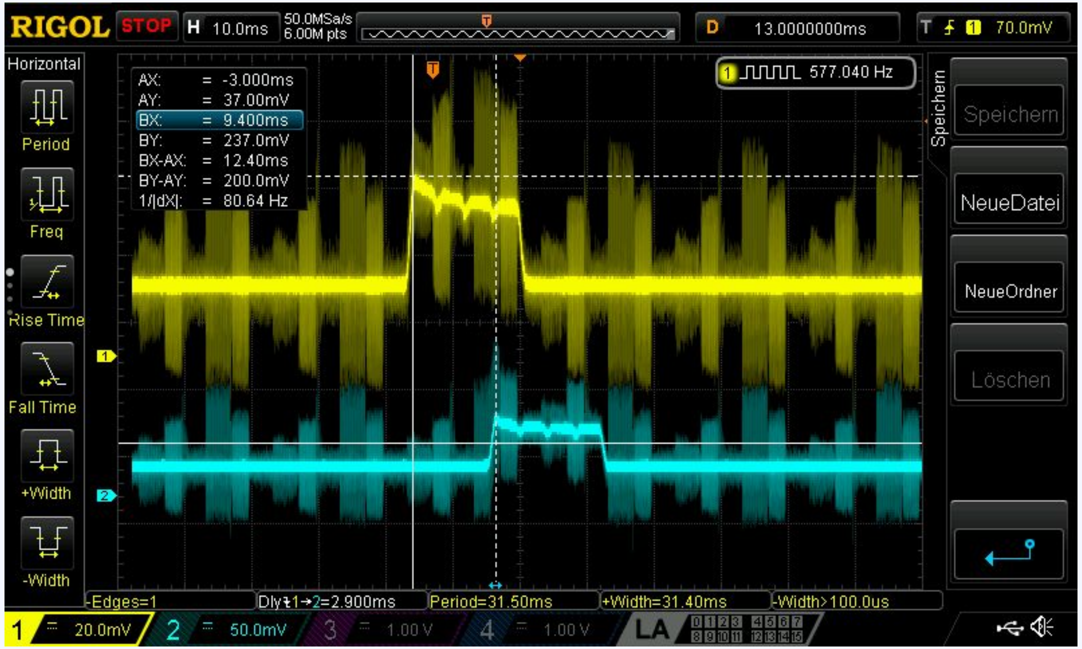
Display scanout

Measurements shows

12,40 ms

60 hz => 16 ms





```
H 10.0ms   50.0MSa/s
           6.00M pts        D 13.0000000ms      T 1 70.0mV

AX:      =  -3.000ms
AY:      =  37.00mV
BX:      =  9.400ms
BY:      =  237.0mV
BX-AX:   =  12.40ms
BY-AY:   =  200.0mV
1/|dX|:  =  80.64 Hz

1 577.040 Hz

-Edges=1   Dly1→2=2.900ms   Period=31.50ms   +Width=31.40ms   -Width>100.0us
1 = 20.0mV   2 = 50.0mV   3 = 1.00 V   4 = 1.00 V
```

Displays are refreshed:

pixel by pixel	right to left
row by row	top to down

inovex

Rendering and vsyncs

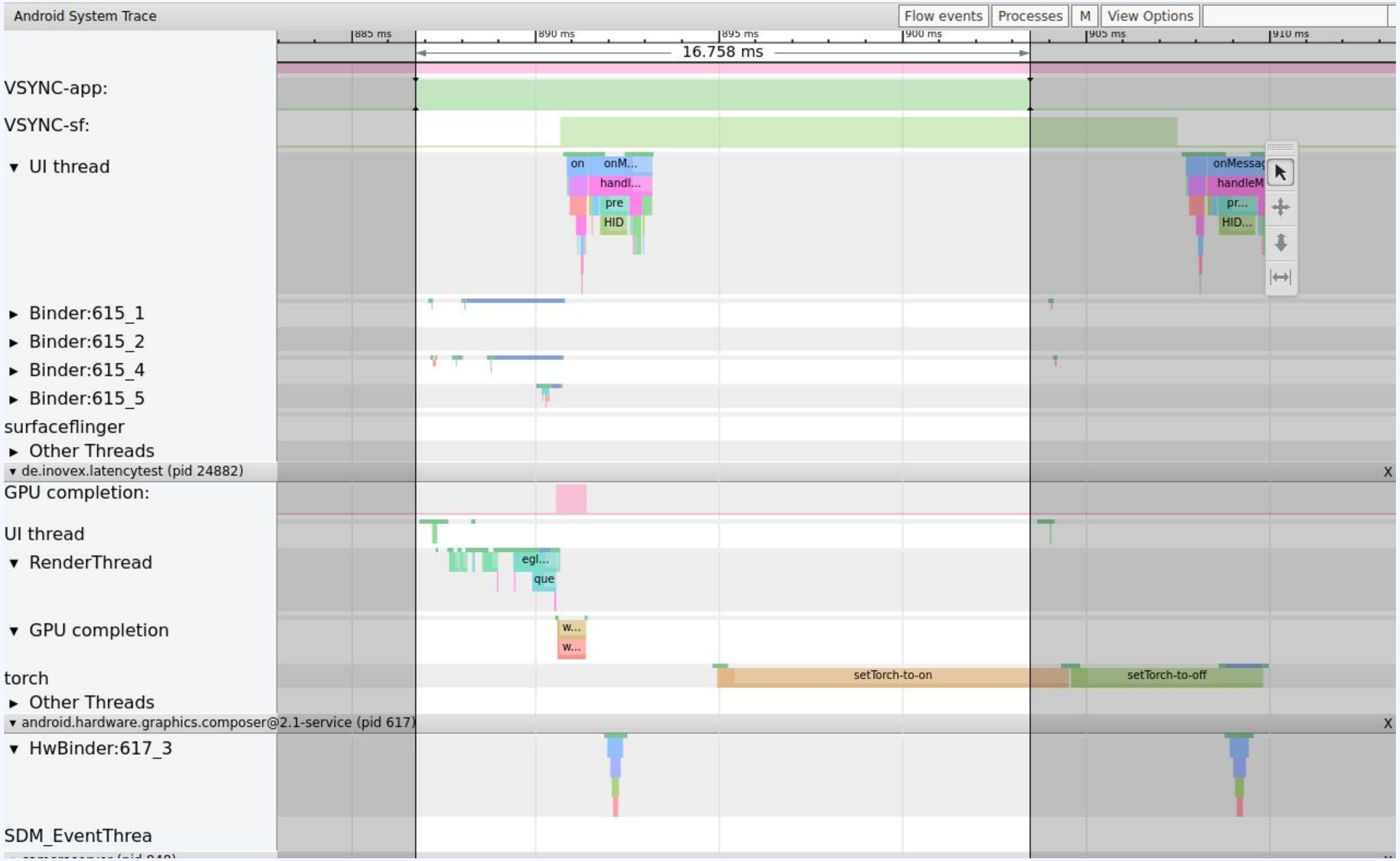
There are three VSYNCs. One hardware and two software VSYNCs:

- HW_VSYNC_0: Hardware starts scanout of next frame
- VSYNC-sf: surfaceflinger starts composition the next frame from the app screen, the status bar, the buttons and overlays.
- VSYNC-App: App starts rendering the next frame based on new input

In every stage of this pipeline a frame is processed:

→ **Tripple buffering**

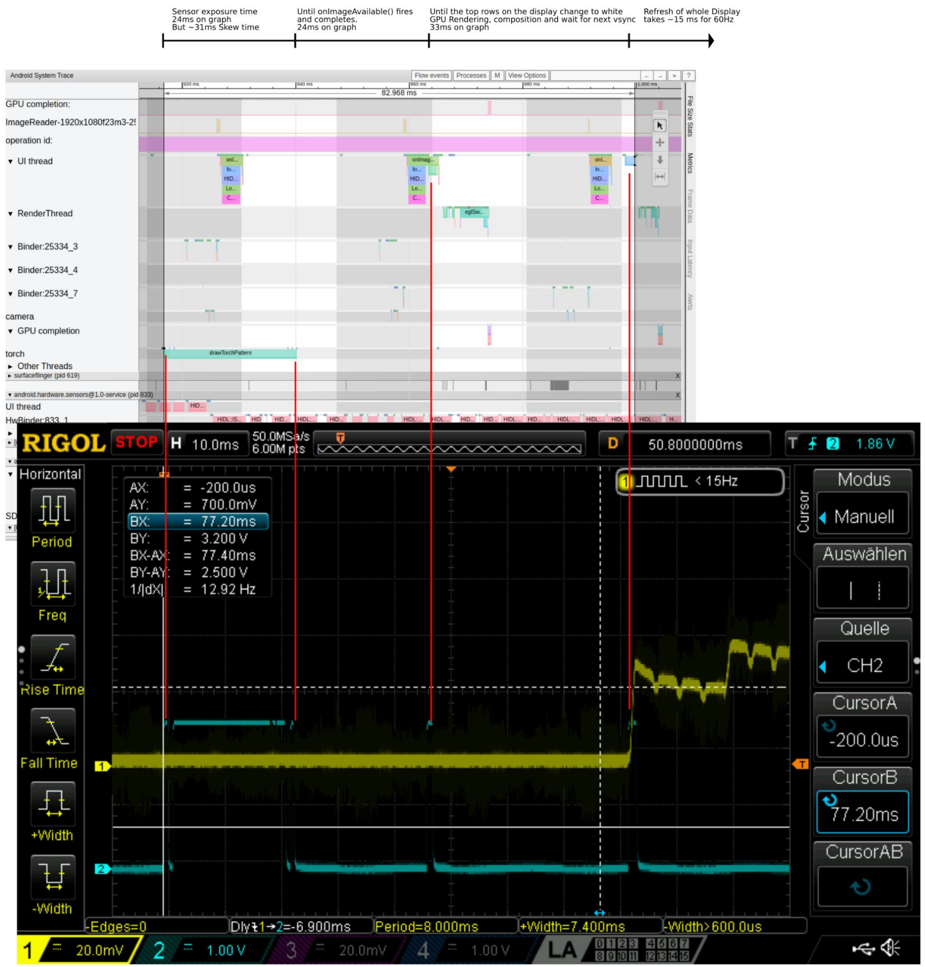
More Infos:
https://source.android.com/docs/core/graphics/implement-vsync


Android System Trace
Flow events
Processes
M
View Options
885 ms
890 ms
895 ms
900 ms
905 ms
910 ms
16.758 ms
VSYNC-app:
VSYNC-sf:
UI thread
on
onM...
handl...
pre
HID
onMessag
handleM
pr...
HID...
Binder:615_1
Binder:615_2
Binder:615_4
Binder:615_5
surfaceflinger
Other Threads
de.inovex.latencytest (pid 24882)
GPU completion:
UI thread
RenderThread
egl...
que
GPU completion
w...
w...
torch
setTorch-to-on
setTorch-to-off
Other Threads
android.hardware.graphics.composer@2.1-service (pid 617)
HwBinder:617_3
SDM_EventThrea

inovex

Finished with the details!

Let’s go back to the first graphics.



```
H 10.0ms   50.0MSa/s 6.00M pts   D 50.8000000ms   T 1.86 V
AX:     =  -200.0us
AY:     =  700.0mV
BX:     =  77.20ms
BY:     =  3.200 V
BX-AX:  =  77.40ms
BY-AY:  =  2.500 V
1/|dX|  =  12.92 Hz
Modus: Manuell   Quelle: CH2   CursorA: -200.0us   CursorB: 77.20ms
-Edges=0   Dly1→2=-6.900ms   Period=8.000ms   +Width=7.400ms   -Width>600.0us
1 20.0mV   2 1.00 V   3 20.0mV   4 1.00 V
```

Graphic shows

0-33 ms	camera sensor
24 ms	camera to app
~33 ms	app to display
0-16 ms	display scanout

→ The latency is 57-106 ms

Recap

Nothing is happening instantly:

- exposure of the different pixels happen at different times
- updating pixels of the display happen at different times

→ The glass to glass latency is a **range**.

It depends on which pixel you light up and measure:

- **first** row vs **last** row of the camera sensor
- **first** pixel or **last** pixel of the display

I love systrace.

It’s the tool to inspect and debug *your* performance issue on Android!

Thank you!

Time for questions!

Stefan Lengfeld
Embedded Software Developer

stefan.lengfeld@inovex.de

Schanzenstraße 6-20
51063 Köln

inovex is an IT project center driven by innovation and quality, focusing its services on 'Digital Transformation'.

- founded in 1999
- 500 employees
- 8 offices across Germany

www.inovex.de

inovex

Further reading and code

Blogposts:

Motion to photon latency in mobile AR and VR by Daniel Wagner

Why is making good AR displays so hard? by Daniel Wagner

Virtual Reality – Blatant Latency and how to Avoid it by Freddi Jeffries

Collection of my tests and code:

https://github.com/inovex/android-glass-to-glass-latency

Blogpost for this talk:

Glass-To-Glass Latency on Android – How Fast Is Your Smartphone's Camera Really? - inovex GmbH

inovex

From smartphones to dashboards: The era of Android Automotive

Title of my talk:
Glass-To-Glass Latency in Android - How to debug the Android Graphics Stack

Abstract:
What is the glass-to-glass latency in Android? It’s the delay between the camera taking a picture and the screen displaying the picture again. This presentation is a technical tour through the Android graphics system and hardware. From the measurement setup, based on cables, LEDs, photodiodes and an oscilloscope, to camera sensors and the rolling shutter effect, to displays and refresh rates, to surfaceflinger and vsyncs, and last but not least, to the ultimate tool to debug most of your performance issues, to Android's systrace.

Link: https://www.meetup.com/inovex-munich/events/299182118/

inovex